ON THE MULTILEVEL SOLUTION ALGORITHM FOR MARKOV CHAINS

Graham Horton*
Computer Science Department
University of Erlangen-Nürnberg
Martensstr. 3
91058 Erlangen, Germany
graham@informatik.uni-erlangen.de

Abstract

We discuss the recently introduced multilevel algorithm for the steady-state solution of Markov chains. The method is based on an aggregation principle which is well established in the literature and features a multiplicative coarse-level correction. Recursive application of the aggregation principle, which uses an operator-dependent coarsening, yields a multi-level method which has been shown experimentally to give results significantly faster than the typical methods currently in use. When cast as a multigrid-like method, the algorithm is seen to be a Galerkin-Full Approximation Scheme with a solution-dependent prolongation operator. Special properties of this prolongation lead to the cancellation of the computationally intensive terms of the coarse-level equations.

*This research was supported in part by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001. The work was also carried out while the author was a Visiting Assistant Professor at the Mathematics and Computer Science Department of the University of Denver, Denver, CO.

1 Introduction

Markov chains describe discrete-state stochastic processes in which the probabilities of transitions between states are a function solely of the current state of the chain – the so-called memoryless property. Since this property is approximately satisified by many physical systems, Markov chains are used widely in stochastic modeling. We will draw our examples in this paper from the performance and reliability modeling of computer systems. It is common to distinguish between continuous time Markov chains (CTMCs), in which transition coefficients between states are interpreted as exponentially distributed rates or delays, and discrete time Markov chains (DTMCs), where they are treated as probabilities. In the latter case, the Markov chain is described by a stochastic matrix. In the steady-state case, CTMC problems can be converted via a simple transformation into problems described by a DTMC. Ultimately, the Markov chain represents a linear system of equations which is usually sparse and often extremely large.

The goal of modeling computer systems is to derive information on performance, measured typically as job throughput or component utilization, and availability, defined as the proportion of time a system is able to perform a certain function in the presence of component failures and possibly also repairs. Various abstract modeling tools for computer systems are in widespread use today, the most important of which are generalized stochastic Petri nets (GSPNs) [1] and queueing networks [6]. When the memoryless condition is satisfied, such models are equivalent to Markov chains, and it is required to solve the Markov chain in order to derive useful information about the abstract model.

Unfortunately, the number of states of the Markov chain (and thus the dimension of the linear system) grows extremely quickly as the complexity of the model is increased. There is one unknown for each state that the model may be in - a number that is subject to a combinatorial explosion. Thus, the Markov chains that have to be solved even for relatively coarse computer models may have tens or hundreds of thousands of states. Apart from their size, one further drawback of typical Markov chains is the presence of coefficients on a wide range of scales. Consider, for example, a reliability model of a computer, in which the rate of component failure may be only once in every few months, whereas the rates associated with the normal behaviour of the system are measured in kHz and MHz.

The resulting large systems of equations must be solved numerically using an iterative scheme. Typical iterative methods in use in the computer modelling community are the Power, Gauss-Seidel (GS), and successive over-relaxation (SOR) algorithms. Surveys of currently used methods may be found in [12, 8]. All of these methods have the drawback that they may require many iterations to reach an accurate solution, particularly if the system is large or if coefficients of strongly varying magnitude are present. This can lead to unacceptably long computation times.

In this paper, we will consider a multilevel (ML) solution algorithm for Markov chains, which was introduced in [5]. The method is based on the principle of iterative aggregation and disaggregation, a well-established numerical solution technique for Markov chains [7, 16, 14]. This principle utilizes a coarse-level level correction that is multiplicative, rather than additive, i.e. newly obtained coarse-level values are used as a factor by which fine-level approximations are rescaled. Furthermore, the aggregation itself, or the coarsening strategy is operator-dependent, in that locally strongly coupled states are mapped together.

Algebraic multigrid (AMG) [13] is considered to be an attractive solution strategy for the systems of equations that are unstructured and which may have strongly varying coefficients. These are characteristic properties of the Markov chains that typically occur in practice. However, as far as we know, until now there has been no AMG approach to solving Markov chains.

It will be shown that the multilevel method is equivalent to an algebraic multigrid scheme which uses the Galerkin method for the coarse level operator and is of Full Approximation Scheme (FAS) type. When viewed as a multigrid scheme, the novelty of the multilevel method is seen to stem from the definition of the prolongation operator, which is solution-dependent and yields the identity operator when combined from the left or from the right with the restriction operator. This has two interesting effects: the right-hand side of the coarse level equations degenerates into a simple restriction of the fine-level right-hand side, and the coarse level operator is solution-dependent and therefore changes from iteration to iteration, even though the Markov chain problem itself is linear.

In the following section, we describe the problem and the aggregation equations. The multilevel method is described in section 3. In section 4, the multilevel method is rewritten and interpreted as a multigrid scheme. In section 5 experimental results for Markov chains arising from a well-known

multiprocessor reliability model and from a simple queueing network are presented, showing the superiority of the method over the standard Gauss-Seidel scheme. The final section summarizes the paper.

2 Problem Description and Aggregation Equations

Consider an irreducible Markov chain consisting of n states $s_1, s_2, \ldots, s_n$. Denote the unknown vector by u, where u_i is the steady-state probability of the Markov chain being in state s_i. In order to facilitate the notation for the multilevel scheme, we use indices l, $l-1$, etc., to indicate levels of aggregation of the Markov chain. The original Markov chain is designated to be at level $l = lmax$.

We then have to solve the system of equations defined by the Markov chain

$$A^l u^l = 0 \ , \tag{1}$$

with the additional condition

$$\sum_{i=1}^{n} u_i^l = 1 \ ,$$

where $l = lmax$. Note that, in the discrete time case, equation (1) is usually written as

$$\pi P = \pi \tag{2}$$

where $\pi = (u^l)^T$ and $P = (A^l)^T + I$, which is the so-called transition matrix. Equation (2) defines the solution of the Markov chain in terms of an eigenvalue problem, and since P is a stochastic matrix, we know that it possesses the unique maximal eigenvalue $\lambda = 1$. We will, however, use the notation of equation (1) throughout, reserving the symbol P for the prolongation operator. In the continuous time case, equation (1) is usually written

$$\pi Q = 0 \ ,$$

where $Q = (A^l)^T$. Matrix A^l has zero column sums and all off-diagonal coefficients are non-negative, making it a singular M-matrix of rank $n-1$.

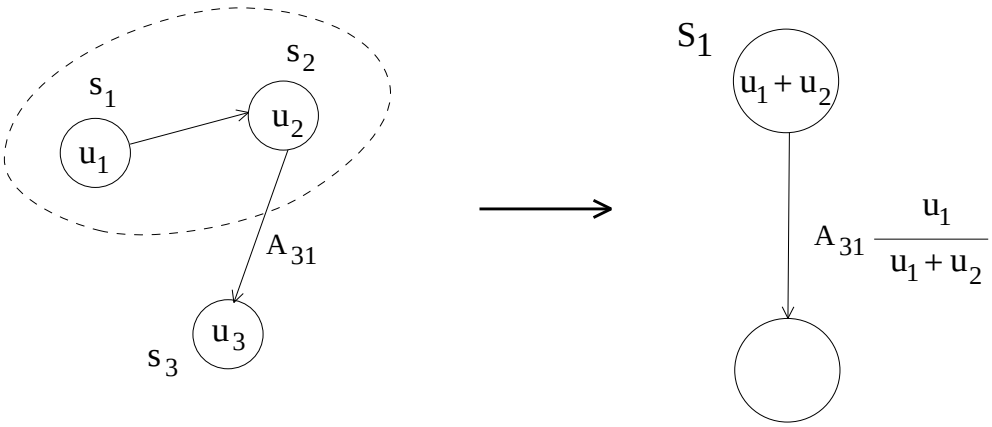

Figure 1: Aggregation of Markov Chains

A coarser representation of the Markov chain described by matrix A^l may be obtained by *aggregation*. This amounts to creating a new Markov chain described by a matrix A^{l-1} with the vector of state probabilities u^{l-1}, each of whose N states $S_1, S_2, \ldots, S_N$ is derived by lumping together a number of states of the original system. Aggregation is motivated by probabilistic arguments which are illustrated in figure 1. The figure shows four states s_1, s_2, s_3, s_4 of a Markov chain, where the probability of the chain being in each of the states is given by u_1, u_2, u_3, u_4, respectively. In addition, a transition from s_1 to s_3 with the probability A_{31} is assumed. The probability of the chain being in either state s_1 or state s_2 is then given by $u_1 + u_2$, and we may replace this pair of states by the corresponding "macro-state" S_1, and similarly for states s_3 and s_4. The transition A_{31} is then mapped to a transition between macro-states S_1 and S_2 with a probability value of $A_{31}u_1/(u_1 + u_2)$, which represents the original transition probability multiplied by the relative probability of the Markov chain being in s_1, given that it is the macro-state S_1. This argument allows us to generate a coarse level (aggregated) system from our original Markov chain.

In the following, we will use the terms *fine level* and *coarse level* to refer to Markov chains, where the latter is obtained by aggregation from the former. The relation $s_i \in S_I$ signifies that the fine level state s_i is mapped by the aggregation operation to the coarse level state S_I. A set of fine states mapped to a common coarse level state $\{s_i : s_i \in S_I\}$ will be referred to as an

aggregate.

The matrix A^{l-1} of the aggregated system is thus defined as follows :

$$A_{IJ}^{l-1} = \frac{\sum\limits_{s_j \in S_J} \left(u_j^l \sum\limits_{s_i \in S_I} A_{ij}^l \right)}{\sum\limits_{s_j \in S_J} u_j^l} \tag{3}$$

This is the classical *aggregation matrix.* Note that the matrix A^{l-1} is a function not only of the fine level matrix A^l, but also of the fine level solution vector u^l. It will be shown in section 4 that this coarse level matrix is equivalent to the Galerkin operator in the multigrid context, with special intergrid transfer operators.

It is well-known that this aggregation strategy propagates the Markov chain property to the coarser level [15], i.e., the matrix A^{l-1} is also an irreducible Markov chain. This yields the aggregated equation in the unknown u^{l-1}:

$$\begin{aligned} A^{l-1}u^{l-1} &= 0 \\ \sum_{I=1}^{N} u_I^{l-1} &= 1 \quad . \end{aligned}$$

It can then be shown that the solution of the coarse system satisfies

$$u_I^{l-1} = \sum_{s_i \in S_I} u_i^l \quad ,$$

i.e., the probability of being in a coarse level state is the sum of the probabilities of being in any of its constituent fine-level states. We will use the aggregation equation as a basis for the multilevel method, whereby we approximate the exact solution values u^l in (3) by values from the current iterate.

The coarse-level matrix depends on the fine-level solution, and must therefore, in the context of an iterative method, be approximated by using the values of the current iterate. More precisely, A^{l-1} is a function of the *relative* values of the fine-level nodes with respect to the values of their aggregates. These are the probabilities of the Markov chain being in a fine-level state conditioned on being in the aggregate state.

The motivation for the multilevel method lies in the observation that, if it is possible to obtain improved values for the relative probabilities of fine-level states in a common aggregate, then an improved coarse-level system can be set up and solved, the solution of which represents the probabilities of the aggregates. The argument can, of course, be applied recursively. We choose to form small aggregates composed of strongly-coupled neighboring states, as the Gauss-Seidel iteration is able to achieve an improvement in the relative probabilities of such states efficiently. We refer to the Gauss-Seidel method in this context as a "smoother", although its role here is somewhat different. This motivation parallels that of multigrid: high-frequency errors are smoothed out on the fine level, whereas low-frequency errors are reduced by the coarse-level correction.

The solution values obtained on the coarser level are used to rescale the values within each aggregate by the same factor. This rescaling guarantees that the fine-level solution remains a probability vector, i.e., the coarse level correction produces new fine-level values in the range $[0, 1]$.

The coarse level Markov chain thus derived forms the basis of the well-known iterative aggregation-disaggregation algorithms [14] in which an aggregate-wise block Gauss-Seidel or block Jacobi iteration on the fine level alternates with a coarse level correction. These methods bear a strong resemblance to domain decomposition methods for partial differential equations.

In contrast to these "two-level" schemes, we will develop a multilevel solution method by recursive application of aggregation where the problem defined at each level represents a Markov chain. The converged solution value at each coarse level state is the sum of the converged solution values of its constituent fine level states. It is to be hoped that similar improvements in performance over single-level iterations such as Gauss-Seidel can be obtained as is the case for multigrid and elliptic partial differential equations. The improvement will come from choosing small aggregates, for which it will be sufficient merely to improve the relative probabilities using a few Gauss-Seidel sweeps, rather than to use large aggregates consisting of many unknowns, and to solve for these values, which can be extremely expensive. In addition, the recursive application of the aggregation will allow us capture relative probabilities at all scales.

3 Multilevel Solution Algorithm

In this section, we recall the recently introduced multilevel algorithm [5], which is based on a recursive aggregation of the Markov chain to obtain approximations of successively smaller dimensions. The algorithm passes through all levels of the hierarchy of chains in a multigrid-style V-cycle. The coarser level equations are the aggregation equations of section 2.

We adopt the following abbreviations for elementwise multiplication and division on vectors in $\mathbb{R}^m$:

$$\begin{aligned} a = b * c \quad &\Leftrightarrow \quad a_i = b_i * c_i, \quad 1 \le i \le m \\ a = b/c \quad &\Leftrightarrow \quad a_i = b_i/c_i, \quad 1 \le i \le m \end{aligned}$$

In the following, $\tilde{u}$ represents an intermediate vector, $\bar{u}$ an approximation to the solution vector, and u^* a correction vector. We denote by $(u^l)^{(i)}$ the i-th iterate. One iteration of the *two*-level version of the ML algorithm, using one relaxation sweep on the fine level, is given by the following sequence of steps.

1. Perform one Gauss-Seidel relaxation sweeps on the finer level, which we denote by
$$\tilde{u}^l = GS((u^l)^{(i)})$$

2. Restrict the current approximation to the solution to the coarse level, where the restriction operator R is defined as summation of the values of fine-level states mapped to a common coarse state:
$$\tilde{u}^{l-1} = R(\tilde{u}^l) \quad \Leftrightarrow \quad \tilde{u}^{l-1}_I = \sum_{s_i \in S_I} \tilde{u}^l_i \ .$$
R can be represented by a $N \times n$ matrix of zeros and ones.

3. Compute the coarse level matrix $\tilde{A}^{l-1}$ as an approximation to that of equation (3) using the current values of the solution vector:
$$\tilde{A}^{l-1}_{IJ} = \frac{\sum_{s_j \in S_J} \left(\tilde{u}^l_j \sum_{s_i \in S_I} A^l_{ij} \right)}{\tilde{u}^{l-1}_J} \ . \tag{4}$$

4. Solve the coarse Markov chain problem for $\bar{u}^{l-1}$:

$$\tilde{A}^{l-1}\bar{u}^{l-1} = 0 \ , \qquad \sum_{I=1}^{N} \bar{u}_I^{l-1} = 1 \ . \tag{5}$$

5. Compute the coarse-level correction as the ratio of new coarse level solution and restricted fine-level solution. In this step, we compute the factor by which the probability of each aggregate must be corrected:

$$(u^{l-1})^* = \bar{u}^{l-1}/\tilde{u}^{l-1} \ .$$

6. Compute the fine-level correction as a prolongation $\bar{P}$ of the coarse-level correction vector. All fine level states of an aggregate are corrected by the same factor:

$$(u^l)^* = \bar{P}((u^{l-1})^*) \quad \Leftrightarrow \quad (u^l)_i^* = (u^{l-1})_I^* \quad s_i \in S_I \ .$$

   $\bar{P}$ can be represented by a $n \times N$ matrix of zeros and ones.

7. Apply the fine-level correction, i.e., rescale the aggregate probabilities to obtain the new $i+1$th iterate:

$$(u^l)^{(i+1)} = \bar{u}^l = \tilde{u}^l * (u^l)^* \ . \tag{6}$$

In this two-level form, the method is similar to the iterative aggregation-disaggregation (IAD) method of Koury, McAllister and Stewart [7], except that a pointwise, rather than a block Gauss-Seidel, approach is used. The multilevel algorithm is obtained by recursive application of the two-level algorithm to obtain a solution to the aggregated equation (5). It is described in algorithmic form in figure 2. The coarse level $l-1$ and fine level l, between which the operators $\bar{P}$ and R map, are identified by appropriate indices. We allow in general the possibility of applying `GS` ν_1 times at each level as a pre-smoothing step and ν_2 times as a post-smoothing step.

The multilevel method is identical in structure – and similar in motivation – to a standard multigrid V-cycle. The principal difference resides in the derivation of the fine-level correction from the coarse level solution vector. Whereas here the correction is a multiplication by the prolonged ratio of new-to-old coarse level solutions, in standard multigrid it is the addition of the

```
procedure ml(l)
   if (l = 0)
      solve A^l ū^l = 0
   else
      ũ^l = GS^{ν1}(ū^l)
      ũ^{l-1} = R_{l-1,l}(ũ^l)
      Compute A^{l-1}
      ml(l − 1)
      (u^{l-1})* = ū^{l-1} / ũ^{l-1}
      (u^l)* = P̄_{l-1,l}((u^{l-1})*)
      ū^l = ũ^l * (u^l)*
      ū^l = GS^{ν2}(ū^l)
   return
```

Figure 2: Multilevel Algorithm

prolonged difference between these two vectors. However, in the following section, by absorbing the current approximation vector into the prolongation operator, we can interpret the ML method as a multigrid scheme.

As is the case for algebraic multigrid [13], we use an aggregation strategy that maps strongly-coupled fine-level states to a common coarse level state. In general, aggregation is pairwise, but aggregates consisting of three or four states, or even as few as one state, are permitted if the coefficients so demand. When the Markov chain contains strongly differing rates - which is somewhat analogous to the presence of strong local convection or anisotropy in PDEs - the convergence rate of the ML scheme is sensitive to the aggregation strategy. We use a greedy algorithm to determine the aggregates whose complexity is linear in the number of edges of the Markov graph.

4 Interpretation as a multigrid method

The multilevel method is based on the iterative aggregation-disaggregation strategy, which dates at least from 1975 [16] and whose equations are derived in a natural way by probability arguments. In this section, we will show that

the scheme can be written as a classical algebraic multigrid algorithm and point out the particular choices of multigrid components that the ML scheme represents.

We begin by considering the multiplicative coarse level correction as the composition of steps 5, 6, and 7 in the algorithm of section 3. This correction is defined by

$$\bar{u}_i^l = \tilde{u}_i^l * \frac{\bar{u}_I^{l-1}}{\tilde{u}_I^{l-1}} \quad s_i \in S_I \ , \tag{7}$$

where the new coarse level solution is used to scale the fine-level solution values in each aggregate by the same factor. The coarse grid correction in multigrid is, however, generally written as an additive correction, so we rewrite equation (7) as

$$\bar{u}_i^l = \tilde{u}_i^l + (\bar{u}_I^{l-1} - \tilde{u}_I^{l-1})\frac{\tilde{u}_i^l}{\tilde{u}_I^{l-1}} \quad s_i \in S_I \ .$$

Thus, the scaling of the fine solution by the prolongation of the ratio of new and old coarse level solutions is equivalent to an additive correction using the prolonged difference between the two coarse vectors scaled by a solution-dependent factor $\tilde{u}_i^l / \tilde{u}_I^{l-1}$. This strategy ensures that the correction step will automatically produce new fine-level solution values that remain bounded in the interval $[0, 1]$. In addition, we observe that the relative probabilities of the states within each aggregate are unaffected by the coarse level correction.

Observation 1 *We may write the ML method with an additive, rather than multiplicative, correction, using the prolongation operator P given as*

$$P = D\bar{P} \ ,$$

where $\bar{P}$ is the standard multigrid prolongation

$$\bar{P} = R^T$$

and D is defined by

$$D = diag\left(\frac{\tilde{u}_i^l}{(R\tilde{u}^l)_I}\right) \ .$$

Thus, the prolongation operator is equivalent to the standard multigrid prolongation operator multiplied by the solution-dependent diagonal matrix D. Strictly speaking, we should therefore represent the dependency of P on $\tilde{u}^l$ by writing $P_{\tilde{u}^l}$, but we generally omit the suffix in the interest of simplicity.

We now make some observations on the thus-defined multilevel algorithm.

Observation 2 *The prolongation and restriction operators satisfy the following conditions:*

$$P \neq R^T \ , \tag{8}$$

$$PRu^l = I^l u^l \ , \tag{9}$$

$$RPu^{l-1} = I^{l-1} u^{l-1} \ , \tag{10}$$

where I^l, I^{l-1} are the identity operators on levels l and $l-1$, respectively.

Proof of (9):

$$\begin{aligned} (PRu^l)_i &= \frac{u_i^l}{\sum\limits_{s_i \in S_I} u_i^l} \sum_{s_i \in S_I} u_i^l \\ &= u_i^l \quad . \end{aligned}$$

□

Properties (8) – (10) are in contrast to the usual case in multigrid. Property (9) has, perhaps, the most interesting consequence, as Observation 5 below will show.

Observation 3 *The coarse level system defined by A^{l-1} from (3) is equivalent to the Galerkin approximation to A^l defined by*

$$A^{l-1} = RA^lP \ . \tag{11}$$

Proof: We may compute the (IJ) coefficient of the matrix RA^lP as the I-th element of the vector RA^lPe^{l-1} where $e^{l-1} = (0, \ldots, 0, 1, 0, \ldots, 0)$ and the 1 is in the J-th position. This gives

$$(RA^lPe^{l-1})_I = \sum_{s_j \in S_J} \left(\frac{u_j^l}{\sum\limits_{s_j \in S_J} u_j^l} \sum_{s_i \in S_I} A_{ij}^l \right) \ ,$$

which is equivalent to (3).

From Observations 1 and 3, we draw the following:

Observation 4 *The ML method is equivalent to a certain FAS Galerkin multigrid iteration.*

Some of the above relationships have been noted by previous authors; Haviv [4] discusses various iterative aggregation schemes, pointing out (9) and (11), and Krieger [8] points out the relationship between IAD schemes and two-level multigrid algorithms.

If we consider applying the equivalent multigrid algorithm to a non singular problem with a non-trivial right hand side, such as a discretized Poisson equation, we obtain the following FAS-Galerkin coarse level equation:

$$RA^lP(\bar{u}^{l-1}) = Rf^l - RA^l\tilde{u}^l + RA^lPR\tilde{u}^l \ , \tag{12}$$

for which we may note the following.

Observation 5 *Property (9) of the ML prolongation operator leads to cancellation of the second and third terms in the right hand side of (12), yielding the vector Rf^l which is constant and may be precomputed:*

$$RA^lP(\bar{u}^{l-1}) = Rf^l \ .$$

In traditional multigrid methods the coarse system is driven by the changing right hand side (the restricted fine-level defect). For linear problems, the coarse matrix is constant and may be precomputed. In the present method, the situation is reversed: the forcing function is constant and successive coarse solutions are driven by a changing system matrix. The computational saving of the latter scheme in the evaluation of the right hand side is substantial: one fine and one coarse matrix-vector product per iteration.

The convergence properties of the ML method will depend on the quality of the approximation of the coarse solution. Since the coarse equations are solution-dependent and vary from iteration to iteration, it is worthwhile to ask how well the coarse matrix (4) approximates the converged matrix (3) and, therefore, how close the computed solution $\bar{u}^{l-1}$ can be to the converged solution u^{l-1}.

Definition 1 *We define an approximation $\tilde{u}^l$ to the exact solution u^l to be smooth, if the relative magnitudes of all solution values within each aggregate*

are in error by no more than $\mathcal{O}(\epsilon)$*:*

$$\frac{\tilde{u}_i^l}{\sum\limits_{s_j \in S_I} \tilde{u}_j^l} - \frac{u_i^l}{\sum\limits_{s_j \in S_I} u_j^l} \leq \mathcal{O}(\epsilon) \qquad s_i \in S_I \ . \tag{13}$$

Since aggregates are composed of neighboring fine-level states, this definition of smoothness is consistent with that of algebraic smoothness in the context of algebraic multigrid [13]. The smoothing property (reduction of high-frequency algebraic errors) is equivalent to achieving approximately correct relative magnitudes of neighboring solution values. Recall that the quality of the coarse level matrix (4) depends only on the relative, rather than the absolute, sizes of the solution values in each aggregate.

Using the eigenvalue problem formulation of equation (1), where $B = (A^{l-1} + I)^T$, we write the coarse problem

$$Bu^{l-1} = u^{l-1} \ . \tag{14}$$

At any iteration of the ML method until convergence is reached, the coarse matrix will differ from its converged value. We write the matrix computed during the ML iteration from $\tilde{u}^l$ as a perturbation $B + \Delta = (\tilde{A}^{l-1} + I)^T$ of the converged value B. The approximate coarse system at any iteration is therefore

$$(B + \Delta)\bar{u}^{l-1} = \bar{u}^{l-1} \ . \tag{15}$$

Note that Δ has the same sparsity pattern as B.

Observation 6 *If (13) is satisfied, then the error matrix* Δ *satisifes* $\Delta = \mathcal{O}(\epsilon)$*. Equations (15) and (14) yield*

$$\bar{u}^{l-1} = u^{l-1} + \mathcal{O}(\epsilon) \ . \tag{16}$$

The coarse solution is therefore also in error by only $\mathcal{O}(\epsilon)$. The exact form of the error term in equation (16) can be found in [9]. We conclude that an algebraically smooth fine-level approximation will yield a coarse system whose solution is an acceptable approximation to that of the converged state.

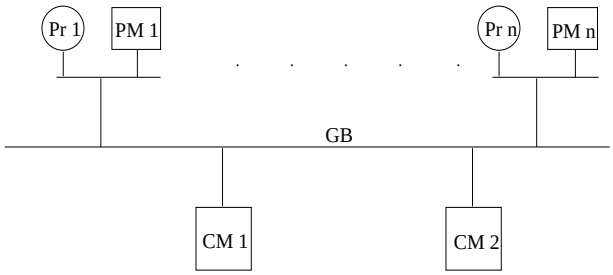

Figure 3: Simple Multiprocessor Example

5 Experimental Results

In computer modelling, Markov chains are seldom developed explicitly; their size alone would in general make this an impractical task. Instead, more abstract modelling paradigms are used, the most important of which are queueing networks and stochastic Petri-nets. In order to demonstrate the gain in efficiency of the ML method over GS, we therefore solve Markov chains defined indirectly via these modelling tools. We choose a stochastic Petri-net model of a multiprocessor from the literature [1], a small queueing network, a well-known queueing network model of a multi-user computing system [15], and a stochastic Petri-net model of a multi-tasking operating system that has recently appeared [3]. The derivation of Markov chains from stochastic Petri-nets is described in [1, 11] and from queueing networks in [6].

Figure 3 shows a multiprocessor system which consists of n processors Pr 1, Pr 2, ..., Pr n, each with a private memory unit PM 1, PM 2, ..., PM n, to which they have direct access via a local bus. The processors communicate via two common memory units CM 1 and CM 2. The processors compete for access to the two common memory units via a global bus GB. Ajmone Marsan, Balbo and Conte [1] give a GSPN model of this multiprocessor (the structure of which is shown in figure 4) which allows for the possibility of failure and repair of the processors, the bus and the memory units. The model allows computation of the loss of effective computational power of the processors due to downtime and competition for the system resources.

Figure 5 shows the computational work of the GS and ML methods applied to this problem, where the numerical values of the parameters are taken from [1]. We show the total number of millions of floating point operations

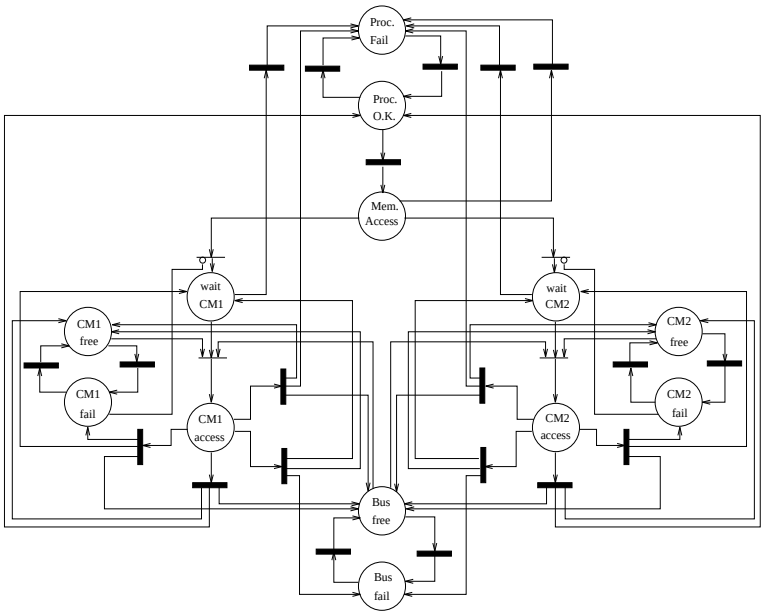

Figure 4: GSPN Model of Multiprocessor System

needed to solve the problem as a function of problem size measured as the number of processors in the model. The number of states of the Markov chains varied from 91 (2 processors) to 3883 (10 processors). All problems were solved to an accuracy of $\|A\bar{u}\|_\infty < 1e-10$.

Comparing the GS and ML schemes, we see that, although these are problems of very small size, the saving in computational effort of ML over GS is quite dramatic: a factor of 27 for the smallest and of 109 for the largest problems considered. It is also clear that the gap widens as the problem size increases.

The SOR method, which is usually used as a solver in software tools for stochastic Petri nets [2, 10], does not improve the situation for this problem, because the optimum overrelaxation parameter is found to be one.

Figure 6 shows a small open queueing system consisting of three single-server queues, as might typically be used in a computer performance model. Server S1 with service rate 90 represents a CPU which receives jobs at a mean rate of 10 from the outside world for processing. There is a 70% probability that jobs leaving S1 may then leave the system. Servers S2 and S3 might represent I/O devices with service rates 7 and 5, respectively. There is a 10%

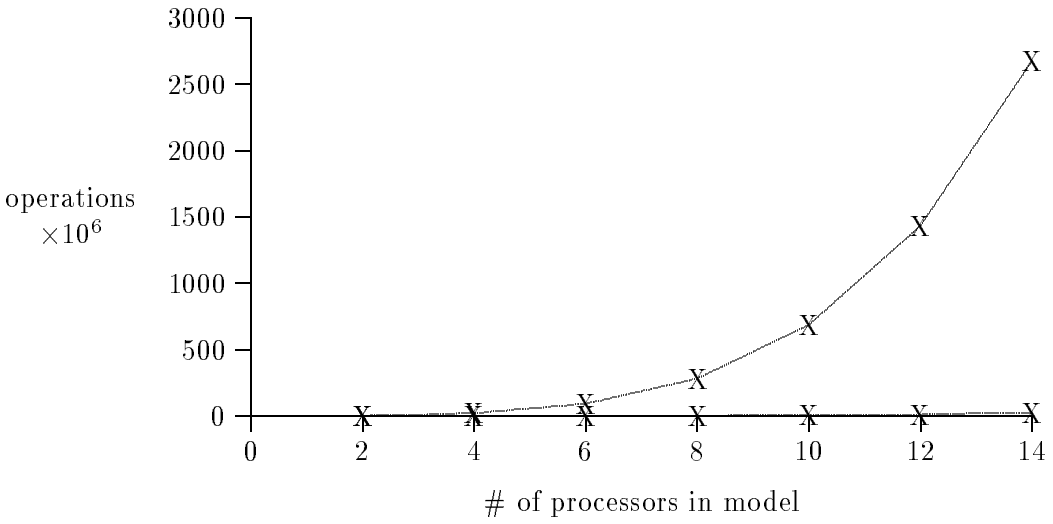

Figure 5: Computational work for GS (upper curve) and ML (lower curve) to solve the Ajmone Marsan/Balbo/Conte problem.

chance that a job leaving S3 returns to S1 for further processing. The queues have a finite capacity c and reject incoming jobs when full. We assume that job arrival times and service rates are exponentially distributed, allowing us to model this system by a Markov chain.

For this problem, the Markov chain has a regular three-dimensional structure which is somewhat analogous to that of a finite-element discretization of a PDE. Each state of the chain may be characterized by the vector $(n_1,\ n_2,\ n_3)$, where n_i denotes the number of jobs in queue i, which may also be interpreted as a coordinate index in the i-th dimension of the three-dimensional "grid" of states. The transitions contained in the chain are the following:

$$\begin{aligned}
(n_1,\ n_2,\ n_3) &\rightarrow (n_1+1,\ n_2,\ n_3)\\
(n_1,\ n_2,\ n_3) &\rightarrow (n_1-1,\ n_2,\ n_3)\\
(n_1,\ n_2,\ n_3) &\rightarrow (n_1-1,\ n_2+1,\ n_3)\\
(n_1,\ n_2,\ n_3) &\rightarrow (n_1-1,\ n_2,\ n_3+1)\\
(n_1,\ n_2,\ n_3) &\rightarrow (n_1+1,\ n_2,\ n_3-1)
\end{aligned}$$

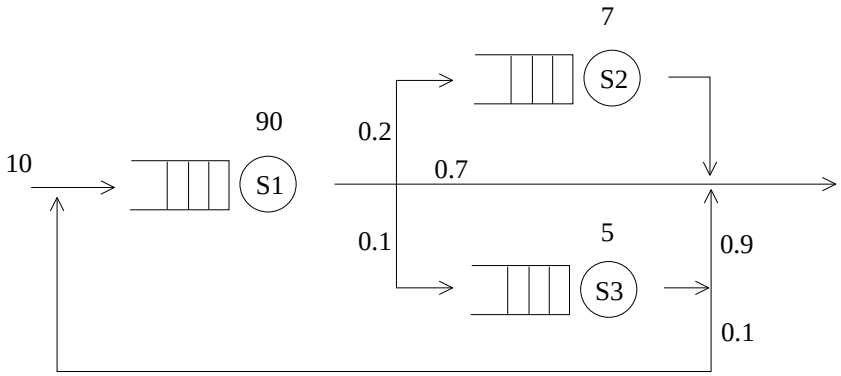

Figure 6: Queueing system test case

$$\begin{aligned}(n_1,\ n_2,\ n_3) &\rightarrow (n_1,\ n_2-1,\ n_3)\\ (n_1,\ n_2,\ n_3) &\rightarrow (n_1,\ n_2,\ n_3-1)\end{aligned}$$

where $0 \leq n_1, n_2, n_3 \leq c$ and transitions to states with negative-valued index are disallowed.

The computational work, measured in millions of floating point operations, required by GS and ML to solve the queueing problem is shown in figure 7. For this problem also, ML is more than ten times faster than GS, although the "sidelength" of the largest Markov chain considered is only 40.

Figure 8 shows the computational effort, measured as millions of floating point operations carried out, for the solution of a computer multiprogramming model which is due to Stewart [15] with the Gauss-Seidel and multilevel methods. The model describes a computing system consisting of a CPU and two I/O devices and the flow of jobs in this system that are initiated by a number of users typing commands at terminals. We used the parameter values as in [15]. This model is of *nearly completely decomposable* type, i.e., it is described by a matrix that is close to block diagonal. The problem is scaled by increasing the number of jobs in the system. Already for 15 jobs, where the Markov chain has 816 unknowns, the multilevel method is a factor of 1083 more efficient than Gauss-Seidel. The improvement grows with the problem size.

Figure 9 shows numerical results for a Markov chain derived from a stochastic Petri-net model of an operating system due to Greiner et al [3]. The model represents the state changes of processes in a computer with a

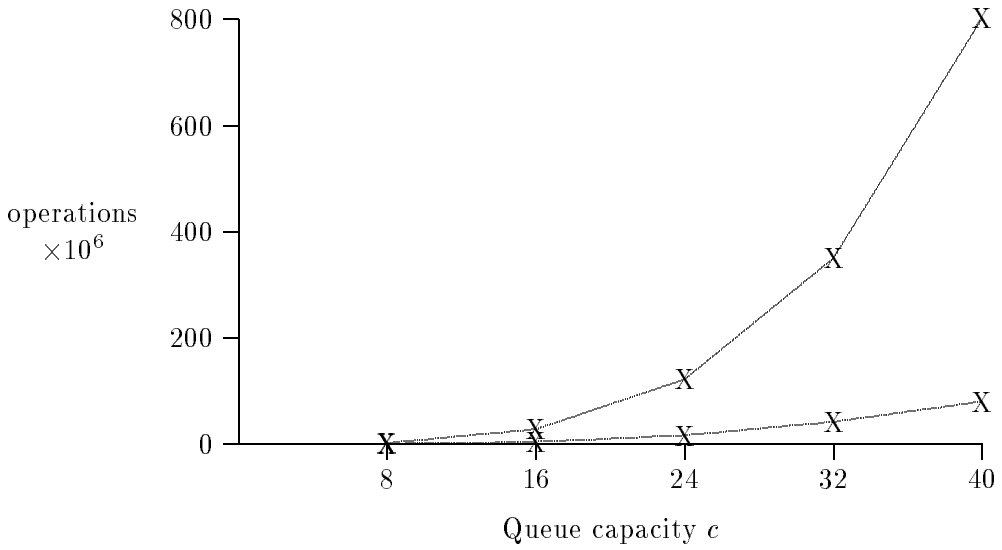

Figure 7: Computational work for GS (upper curve), ML (lower curve) to solve the queueing problem.

multitasking operating system. The problem can be scaled by increasing the number of jobs that are in the system simultaneously. On the left, the computational effort of GS and ML, measured in millions of floating point operations, is shown as a function of problem size. The performance improvement of ML is once again seen to increase with problem size; at 8 jobs ML is 70 times faster than GS. On the right side of Figure 9, the convergence factor is shown. The factor for ML is a constant 0.16 for all problems other than the smallest, whereas that of GS deteriorates from 0.975 to 0.998 over the interval.

6 Conclusions

We have discussed the recently introduced multilevel solution method for the steady state analysis of Markov chains, an important class of problems in the stochastic modelling of physical systems. The method is motivated as a generalization of well-known iterative aggregation-disaggregation techniques

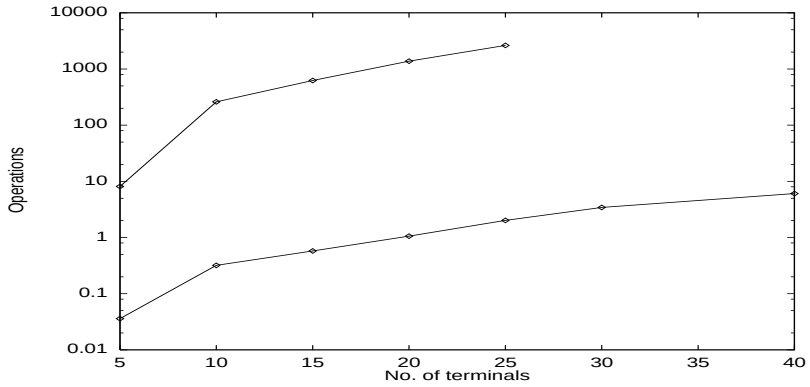

Figure 8: Numerical Results for Interactive Multitasking Model. Upper curve: Gauss-Seidel; Lower curve: Multilevel.

to include multiple levels.

It is shown that the algorithm is equivalent to an FAS multigrid method with a Galerkin-style coarse grid operator and a solution-dependent prolongation operator. The latter property of the scheme ensures that the coarse grid correction produces solution values that are bounded between zero and one and that it leaves the relative probabilities within aggregates unchanged. This may prove useful for the solution of PDEs with similar restraints on the solution, including, for example, mass fraction problems, where traditional multigrid coarse grid corrections may produce under- and over-shoots.

The algorithm is shown to perform well compared to currently used algorithms, obtaining performance improvements of up to three orders of magnitude on the problems selected.

Further work will include a more "multigrid-like" coarsening strategy and prolongation operator which is more similar to an interpolation. In this manner, it is hoped that the poor performance of the multilevel method for homogeneously structured problems with very smooth solutions, which has been observed, can be improved.

7 Acknowledgements

The author acknowledges the collaboration with S. Leutenegger of ICASE in the development of the multilevel scheme. He would like to thank D. Keyes,

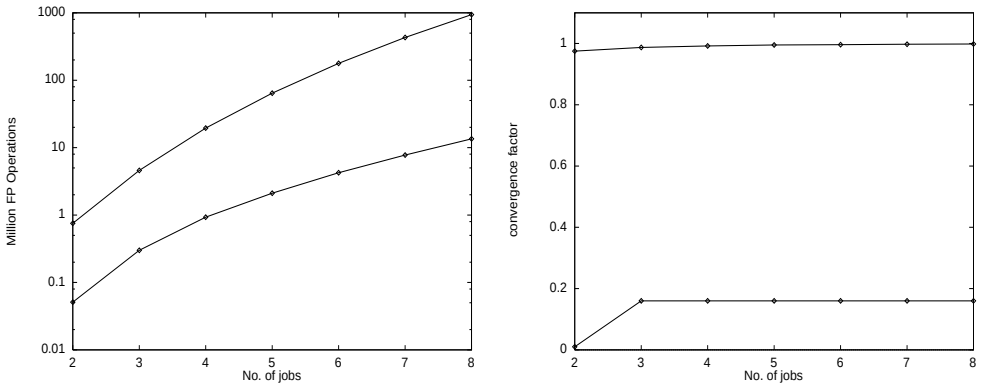

Figure 9: Numerical Results for Operating System Model, Gauss-Seidel and Multilevel methods; Left: Operation count; Right: Convergence factors.

D. Mavriplis and V. Venkatakrishnan of ICASE and M. Holst and S. Vandewalle of CalTech for fruitful discussions. S. McCormick of the University of Colorado at Boulder made several helpful suggestions for the presentation of the material.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte: *Performance models of multiprocessor systems*. MIT Press, 1988.

[2] G. Ciardo, K. Trivedi, J. Muppala: *SPNP: stochastic Petri net package*. Proc. of the Third Int. Workshop on Petri Nets and Performance Models (PNPM89), Kyoto, Japan, Dec. 1989, pp. 142-151. IEEE Computer Society Press.

[3] S. Greiner, A. Puliafito, G. Bolch, K. Trivedi: *Performance Evaluation of Dynamic Priority Operating Systems*, Proc. of Petri-Nets and Performance Models (PNPM95), Durham, NC, Oct. 1995, pp. 241-250.

[4] M. Haviv: *Aggregation/Disaggregation methods for computing the stationary distribution of a Markov chain*. SIAM J. Numer. Anal., Vol. 24, No. 4 (Aug 1987).

[5] G. Horton, S. Leutenegger: *A Multilevel Solution Algorithm for Steady-State Markov Chains.* ICASE Report #93-81, NASA CR-191558, NASA Langley Resarch Center Hampton, VA, September 1993. *And* SIGMETRICS '94 (Proceedings), Nashville, TN, 16th-20th May 1994.

[6] L. Kleinrock: *Queueing Systems, Volume 1: Theory.* Wiley & Sons, 1975.

[7] R. Koury, D. McAllister, W. Stewart: *Methods for computing stationary distributions of nearly completely decomposable Markov chains.* SIAM J. Alg. Disc. Math. Vol. 5, No. 2 (1984), 164-186.

[8] U. Krieger: *Numerical Solution Methods for Large Finite Markov Chains* in R. Marie, G. Harig, G. Kotsis (Eds.) *Performance and Reliability. Performance Evaluation*, R. Oldenbourg, Wien, 1994, 267-318.

[9] P. Lancaster: *Theory of Matrices.* Academic Press, 1969.

[10] C. Lindemann: *DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets.* Performance Evaluation, July 1994.

[11] M. Molloy: *Performance analysis using stochastic Petri nets.* IEEE Trans. Comp., Vol. 31, No. 9, 913-917, Sept. 1982.

[12] B. Philippe, Y. Saad, W. Stewart: *Numerical Methods in Markov Chain Modeling.* Operations Research, Vol. 40, No. 6, Nov. 1992.

[13] J. Ruge, K. Stüben: *Algebraic Multigrid.* In S. McCormick (Ed.) *Multigrid Methods.* SIAM, 1987.

[14] P. Schweitzer: *A Survey of Aggregation-Disaggregation in Large Markov Chains.* In W. Stewart (Ed.) *Numerical Solution of Markov Chains*, Marcel Dekker, 1991, ISBN 0-8247-8405-7.

[15] W. Stewart: An Introduction to the Numerical Solution of Markov Chains, Princeton, 1994.

[16] Y. Takahashi: *A Lumping Method for Numerical Calculations of Stationary Distributions of Markov Chains,* Research Report No. B-18, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1975.